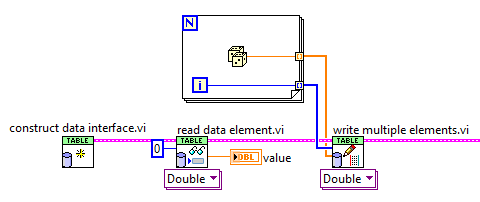
Tag Bus

Control applications frequently need to provide a mechanism for passing single point data between code modules. In many cases, this need is met by a library like the [CVT](https://decibel.ni.com/content/docs/DOC-36858). However, it is sometimes helpful to have a data access mechanism like the CVT while maintaining the performance, traceability, and access restrictions which come from keeping things on the wire. This tag bus library attempts to fill this need, and as such it is most easily thought of as "CVT-on-the-wire".

 Like the CVT, the library is composed of a few separate pieces, the data table, data dictionary, and the tag.

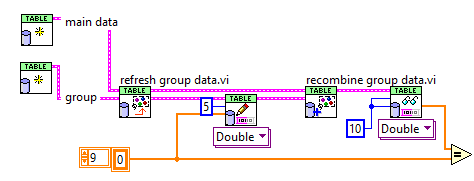
# Data Table

The actual dataset is called the "tag table", "data table" or more generally the "tag bus". In the CVT, this is implemented as a functional global, making it a singleton which can be accessed from anywhere in the system. The tag bus is a wire with a similar structure (arrays of the common data types) and an API to access these elements. The current API includes the common access modes--read/write 1x, r/w sequence (0,1,2,3), and r/w set (1,4,7,34).

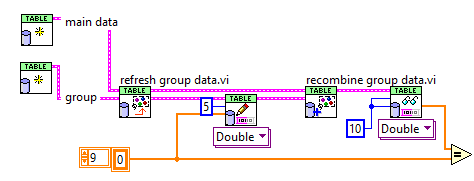


This wire is, unfortunately, a cluster. This is for performance reasons. Originally the tag bus was a class, with private data. However, due to some internal performance issues, classes are significantly slower on the write operation as compared to a cluster. While the overall performance is still significantly higher than the CVT, we decided to err on the side of higher performance. This performance issue is tracked by CAR 457851.

 In addition to the standard reads and writes, it provides two new concepts which are derived from the CVT group concept introduced in version 3. The first is the concept of a group, which is a new tag bus which is a subset of another tag bus. You can specify the tags that are part of the group and copy data from one bus into the other and back again:

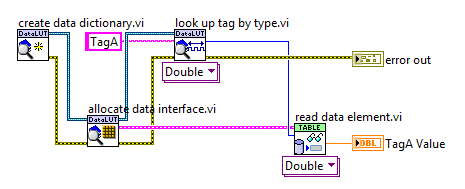


The other concept is a "transfer specification", which basically defines a transfer of certain tags from one tag bus to the other. These tag bus wires do not have to be related, unlike the group. A group transfer is basically a very special case of the transfer specification. A simple example is shown here:



# Data Dictionary

The data dictionary class is the lookup mechanism for the library. In the CVT, this is another FGV shared across every data type. In the tag bus, this is a class we've called the "dictionary". This dictionary takes a tag name and tells you where in the dataset it is located. As shown in the previous picture, it is not necessary to have a dictionary in order to use the tag table, but it allows developers to name values and handle change more easily.

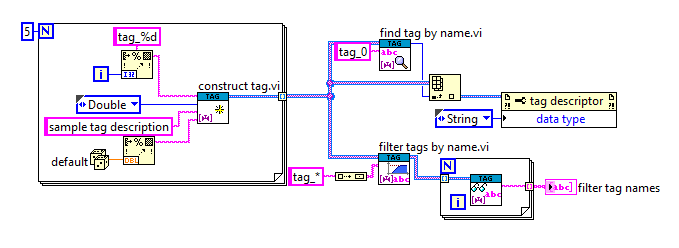


Internal to this lookup, and part of the reason this is beta, is that the lookup uses a new search. Rather than using the traditional variant attribute lookup, this implements a custom binary search. While about 30% slower to initialize, the lookup only takes about half as long as the variant lookup. With this kind of performance boost, it makes it that much more likely that users will be able to do the lookup inside of their performance-critical code, rather than taking steps required in the CVT like looking up data indices during initialization.

# The Tag

The final concept is the tag bus tag class. In the CVT, this was just a simple cluster--which worked well. For our use case, we wished to be able to associate extra data with a tag by creating child classes. For example, you may wish to create a IOV-based tag which uses the base implementation to store a name ("pressure"), type (double), and description ("pressure at the measurement point"), while also storing the associated IOV (ni.var.io://localhost/mod1/AI0). In most code using the CVT, this additional data is either hacked into the description or stored in a separate array which must be updated in parallel to the tag list. Using a class for this configuration information makes APIs like the CIE much easier to configure.

 In addition to being a data storage mechanism--its primary purpose--we've also added a few basic filtering functions.



# Conclusions

This API is currently fairly barebones and may not be suitable for many users. Unlike the CVT, which hides many of the details of the implementation to provide a simple "read tag" function, a tag bus user must explicitly perform the lookup from the data dictionary and then the read from the data table. However, this library has similar power and functionality to the CVT while allowing for higher performance and a local, wire-defined scope.